SPIder Koerier

Lente Editie 2009

Maart 2009
Nummer 2009-1

www.st-SPIder.nl

Redactioneel

Het schaatsen op natuurijs ligt al weer achter ons. Niet alle bestuursleden zijn breukvrij gebleven, maar de schade is reeds weer aardig hersteld. Ik hoop dan ook dat de lezers van de SPIder Koerier met name hebben genoten van de ijspret. Van ijspret naar leespret. In dit nummer hebben een aantal auteurs de kans gegrepen om hun ervaringen te delen met hun SPIder vrienden. Zoals het ijs een breed publiek kon vermaken, van glad pak op de klapschaats tot mutsdrager op het botje, zo biedt deze Koerier ook een variëteit aan onderwerpen: introductie van Agile, kwaliteit, testen, een kijk op vakliteratuur tot ervaringen met een cursus. We kunnen geen chocolademelk digitaal versturen, maar we willen je graag opwarmen met mooie verhalen en informatie die er toe doet. Veel leesplezier!

René Krikhaar

Inhoudsopgave

Van de voorzitter, maart 2009

Het voelt nog wat onwennig om onder het kopje “Van de voorzitter” een stukje te schrijven. Na twee jaar met veel plezier het penningmeesterschap te hebben vervuld, ben ik per 1 januari 2009 tot voorzitter gekozen. De afgelopen periode heb ik met enorm veel plezier met Ben samengewerkt en het is dan ook bijzonder jammer dat hij in 2008 besloten heeft het voorzitterschap neer te leggen. Zijn overwegingen zijn overigens meer dan overtuigend. Daarnaast laat Ben iets achter waar hij met recht trots op kan zijn.

De vraag is nu natuurlijk wat je als nieuwe voorzitter (of eigenlijke nieuw bestuur) gaat doen. Ga je de boel omgooien of ga je op de winkel passen? Eigenlijk beide. We willen op de winkel passen en deze nóg aantrekkijker maken. In dat geval ontkom je er niet aan om dingen te veranderen. En dat is natuurlijk ook goed. Zonder verandering, geen vooruitgang en dat is wel wat we willen.

In het laatste kwartaal van 2008 heeft het bestuur versterking gekregen van twee nieuwe leden; Niek Pluijmert en Wil Leeuwis. Gezamenlijk met een andere voorzitter is het bestuur danig gewijzigd. Iedereen moet zijn plek en weg vinden in deze nieuwe constellatie. Dit heeft overigens zeer snel plaatsgevonden. Het moest haast wel, want je kunt niet te lang een interne focus houden. De winkel moet open blijven. De eerste plenaire sessie is alweer gepland (4 maart aanstaande) en de voorbereidingen voor de conferentie zijn in volle gang.

Het is nu aan het nieuwe bestuur om de streep in het zand weer een stukje verder te trekken. Om dit te kunnen doen, zal het globale kader helder moeten zijn. Aan de ene kant heb je de combinatie van instrumentele modellen (CMMi, Agile, Six sigma, etc.), hoe je dit gaat tailoren, implementeren en borgen (verandermanagement) en wat al deze acties voor de organisatie gaat opleveren; het bedrijfseconomisch perspectief.

Aan de andere kant wil je dit benaderen vanuit het gezichtspunt van de supply kant, de demand zijde en de wetenschap. Dit alles ten aanzien van software process improvement en kwaliteit, waarbij we jullie intensief willen betrekken. Binnen dit globale kader zijn we onze strategie 2009/2010 aan het definiëren.

Zijn we te ambitieus? Ik zal het niet weten. ‘*Reach for the stars and you never end up with a hand full of dirt’*, heeft iemand mij ooit eens verteld. Daar zit een waar woord in.

Kunnen we dit als bestuur alleen? Nee, absoluut niet. Als bestuur kunnen wij het alleen optimaal faciliteren. Om het uiteindelijke doel te bereiken hebben we onze leden en sponsoren nodig. In essentie is SPIder opgericht om kennis te brengen en kennis te halen. Met ruim 1.400 leden en vijf grote sponsoren heeft de Stichting SPIder een enorm en zeer waardevol netwerk. Wanneer dit netwerk volledig ontwaakt krijgen we als participant de beschikking over een schat aan kennis en ervaring waar we allemaal enorm mee geholpen zijn. *We all benefit!*

Ik wil iedereen dan ook oproepen een bijdrage te leveren. Hoe groot of klein deze ook is, het zal altijd waardevol zijn. De bijdrage kan op verschillende manieren plaatsvinden. Artikelen voor de Koerier, bijdrage in plenaire sessies en de conferentie, deelname in werkgroepen of het opstarten van nieuwe werkgroepen zijn hiervan enkele voorbeelden. Het aloude adagium: "*Wie kennis niet kan delen, kan deze ook niet vermenigvuldigen*" is wellicht een open deur, maar daarmee niet minder waar en zeer actueel.

Tot slot wens ik iedereen een bijzonder succesvol en bovenal gezond 2009 en ik hoop je te ontmoeten op één van de komende SPIder bijeenkomsten.

Jeroen Macke
Voorzitter Stichting SPIder

SPIder Conferentie 2009

Op 6 oktober 2009 zal in Ede de twaalfde SPIder conferentie worden georganiseerd. Het thema dit jaar is:

SPInoveren, inSPIreren en tranSPIreren

Veel bedrijven maken een moeilijke tijd door, als gevolg van de kredietcrisis. In het gunstigste geval wordt er sterk in kosten gereduceerd, andere moeten reorganiseren. Extra tijd en budget voor verbeteren is bij veel organisaties voorlopig in de ijskast gezet. Dat wil zeker niet zeggen dat er geen verbeteringen meer kunnen worden gerealiseerd. Het vergt meer creativiteit door elkaar op de werkvloer te inspireren. Procesverbeteringen kunnen nog steeds worden gerealiseerd door er wat extra energie in te stoppen: transpireren.

Deze conferentie staat in het teken van innovatieve oplossingen om processen op peil te houden of te verbeteren. We hebben daar ludiek de term SPInoveren aan gekoppeld. Wat kunnen we als proces verbeteraars toevoegen om de crisis goed door te komen. In**SPI**reren, tran**SPI**reren of **SPI**noveren?

De SPIder conferentie is jaarlijks ook het ideale ontmoetingsmoment om met vakgenoten te spreken over proces verbeteren. Naast inSPIrerende lezingen zal er voldoende gelegenheid zijn om met collega's te spreken. Kortom, een goede investering om inSPIratie op te doen.

Sprekers gevraagd

Wil je spreken op de SPIder conferentie? Schrijf dan in het kort je verhaallijn op (A4) met de belangrijkste conclusies. De precieze procedure staat binnenkort beschreven op de website. De deadline staat op 31 maart 2009.

De laatste nieuwtjes en feiten worden gecommuniceerd via de website:
http://www.spiderconferentie.nl/

C o l u M n

Een stedenbouwkundige vertelde mij laatst dat het erg lastig is om de juiste wetgeving en bijbehorende documenten terug te vinden bij het ontwerpen van bijvoorbeeld een nieuwe woonwijk. De afstand tussen lantaarnpalen of de breedte van de weg worden blijkbaar per gemeente vastgesteld. Dan speelt ook nog een rol welke regels er golden op het moment dat er besloten werd dat er een nieuwe wijk werd gebouwd. Het was zeer lastig om dat terug te vinden (er is geen centrale locatie waar deze informatie is vastgelegd) en zeker als er terug in de tijd moest worden gekeken. Alle (historische) documenten moesten worden doorgenomen om te bepalen welke regels hij moest toepassen. Vanuit een hele andere wereld werd ik geconfronteerd met versiebeheer en veranderingen op documenten. Feitelijk was er gebrek aan een centraal archief (dan moet je denken aan een archief voor heel Nederland), maar ook de baselines ontbeerden. Belangrijke informatie, aan verandering onderhevig, werd niet onder configuratie management gebracht zoals we dit in de software wereld kennen. Ik vertelde natuurlijk stoer het verhaal hoe we dit in softwareontwikkeling onder controle hadden.

Natuurlijk is dat niet overal even waar. Momenteel ben ik samen met een aantal bedrijven bezig om software configuratie management (SCM) beter onder controle te krijgen. Hoewel de theorie reeds lange tijd stabiel is, blijkt de vertaalslag naar de praktijk zeer moeilijk. Identificatie, control, status accounting en auditing leren we uit en lezen we in de SCM boeken. Toch zien we in weinig organisaties dat configuratie management met voldoende volwassenheid aanwezig is. Dit hangt natuurlijk helemaal van de organisatie en het product af. Volgens de verschillende procesmodellen, zoals CMM, is reproduceerbaarheid de eerste stap die men zou moeten realiseren.

Bij het uitbrengen van een software release weet je op voorhand dat deze ooit zal gaan wijzigen. Ofwel er wordt een probleem geconstateerd ofwel een uitbreiding van de functionaliteit is wenselijk. Het is dan cruciaal dat de juiste versie van de broncode kan worden getraceerd die behoort bij de uitgebrachte software release. In deze versie van de broncode kan de wijziging worden aangebracht en de applicatie kan opnieuw worden gebouwd. Het resultaat kan worden getest en bij goedkeuring worden geleverd aan de gebruikers. In dit korte scenario staan de activiteiten beschreven die minimaal (“bare minimum”) moeten zijn ingericht om software ontwikkeling op orde te hebben.

De stedenbouwkundige heeft gelijksoortige problematiek: welke release van wetgeving gebruik ik? Kan ik hierop een wijzigingsvoorstel indienen (de gemeente kan een uitzondering goedkeuren)? Hoe wordt deze goedkeuring vastgelegd (nieuwe release) en kan ik die later weer terugvinden (de identificatie van de juiste release)?

René Krikhaar

Enterprise Class Agility at Ericsson

Jasper Goos and Alfo Melisse
Ericsson
jasper.goos@ericsson.com, alfo.melisse@ericsson.com

Based on "An Ericsson Example of Enterprise Class Agility", by Jasper Goos and Alfo Melisse which appeared in the conference proceedings of the Agile 2008 conference (ISBN 978-0-7695-3321-6) © 2008 IEEE

Abstract

Agile practices have gained momentum within Ericsson[1]. This report describes the experiences of a transition towards Agile in a product development unit (PDU) of about 300 people. This is just a small part of the large Agile roll out in Ericsson's development organization for multimedia products. This paper describes the approach, results, choices made, lessons learned and next steps.

1. Introduction

Early 2007, the vice president of Ericsson's large Development Unit (DU) for multimedia products (2000+ people), initiated an immediate transition to applying Agile practices and mindset. Ericsson R&D in The Netherlands is one of the sites to bring about this transition. This paper is an experience report for this site only.

2. Why Agile

Our DU addresses a very dynamic market in which new business models emerge as well as new players. To be able to compete against new competitors we had to increase our responsiveness. Therefore Ericsson set a corporate R&D goal to reduce the Time to Market (TTM) by 50% in four years time.
To achieve this Ericsson introduced a 'streamline' concept. Part of that concept was timeboxing. The average timebox became about three months instead of six to twelve months. This forced the PDU to deliver products in much shorter cycles. Other improvements were reducing the amount of scope changes by setting the scope just before the timebox starts and ensuring high quality deliveries to end-to-end testing. The results were impressive:

1) Waste reduction: an average reduction from 25% requirements cancelled during projects to 9%.
2) Time to Market reduction: average TTM reduced to around 50% of the 2005 average.
3) Reduction of costs for product maintenance of 20% (less Faults Slip Through).

With the 'streamline' concept in place, DU management saw using Agile practices as a natural next step to continue to improve. Being used to the 'streamline' concept and its principles, the move towards the Agile principles was a logical one.

[1] Ericsson is a provider of technology and services to telecom operators. Ericsson supplies communications services and manages networks that handle more than one billion subscribers worldwide. The company's portfolio comprises mobile and fixed network infrastructure and broadband and multimedia solutions for operators, enterprises and developers. The Sony Ericsson joint venture provides consumers with feature-rich personal mobile devices. Ericsson has more than 75,000 employees and is headquartered in Stockholm, Sweden.

3. The Change Approach

3.1 The Approach

A group of Agile practices in our PDU were applied first in one development project and were introduced in a very short time frame. An Agile consultant gave a one day training and helped with a one day kick-off including a planning game. The project had a fixed scope and fixed delivery time, as the customer contract was already signed. After several iterations the first delivery to the customer was done successfully!
Another Agile consultant, that acted as Agile coach, accompanied the introduction of Agile practices and mindset for a long period (>one year). The practices included a planning game, daily stand-up meeting, cross functional teams, pair programming, short development cycle (two or three weeks), demonstration to "customer", and retrospectives for continuous learning & improvements.
Based on an evaluation of Agile experiences from multiple PDU's, the feedback from the pilot and management observations, a way forward was determined. The way forward is as powerful as it is simple: "continue Agile where it works well and where people are motivated, apply other continuous improvement techniques everywhere else".
Other elements of the way forward are:

- Several new cross-department pilots will be started.
- Improve in the area of getting real customer contact and experiment with synergistic contracts.
- Create more self organizing teams.
- Focus more on the value streams [1] and the biggest opportunities for improvement.

3.2 Team Impact

Before the introduction of Agile the PDU followed a strict waterfall process. The different expertise like Design, Function Test, System Test and Integration & Verification (I&V) were grouped in separate departments.
Nowadays project teams are multidisciplinary and consist of all functions they need: department borders are being crossed. Everybody works together from the beginning of the project.
The impact on the teams is huge. Besides the new team composition and new practices the teams received a much broader responsibility: they are self-managing, estimate hours for the different engineering tasks, make an iteration planning and commit to it.
A number of roles, both inside and outside the team, have changed. One of the team members is acting as a team coach (Scrum Master). Outside stakeholders (like the product owner and the software architect) work with the team and not with a representative. The customer is represented by the product owner, who is in fact a product manager.
The team members were "taught" the Agile concepts and practices. By giving the teams new responsibilities, the motivation to continue with the practices came naturally. Disciplined practice execution led to motivation.
Every cultural change causes resistance and Agile is no exception on that. For example working cross-functionally is not always encouraged by everybody.

Knowledge has to be shared more, which is highly appreciated by less senior engineers, but puts a burden on specialists. In the beginning team members were a bit frightened to provide estimates. Some people were scared not to finish the committed user stories in time and to be judged afterwards. Experience, competence and mutual trust gradually increased and took fears away. After a couple of iterations, the skeptics became addicts and the agile process was driven by team motivation. The teams came with new ideas and took new initiatives. People have to get used to Agile and when they notice teams using Agile making such strong progress, they engage more.

4. Results and Figures

4.1 Customer Satisfaction

As customer involvement was still according to traditional style contracts, product management was considered the customer to satisfy. Product management was very pleased with the introduction of Agile practices, they mentioned:

- The Agile practices deal better with the reality of an unpredictable market and necessary scope changes along the way.
- Transparency, timelines and improved requirement clarification increased control on how the product evolved.
- Team commitment was better and motivation increased.
- Less surprises, more influence.

4.2 Time-to-market

One of the reasons to apply Agile was to reduce the time-to-market (TTM). Before we started using Agile we already managed to reduce the TTM by 50%. Now, after more than one year doing Agile, the TTM dropped another 13%, so 63% in total. Main reasons are timeboxing and being able to develop and test at the same time.

4.3 Impact on Quality

Fault Slip Through (FST) is a metric that measures faults that surface after software delivery and that should have been prevented or identified before delivery. A key question with our new practices is how Faults Slip Through has been influenced.

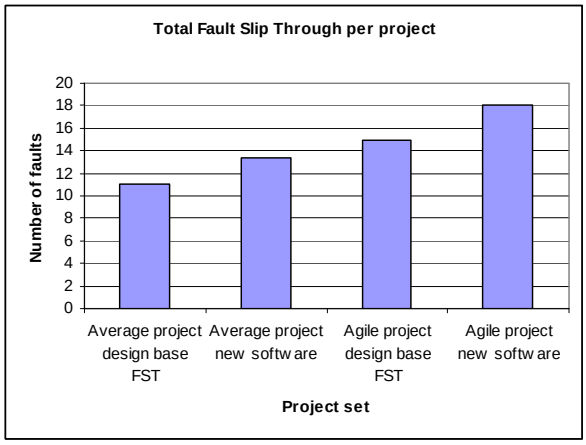


Fig. 1 Fault Slip Through Chart 2007

Figure 1 shows the amounts of faults slipped through for the agile project and for the average project. FST is shown both for faults that have slipped caused by previous product releases (design base faults) and for faults introduced in the new software. FST of the Agile pilot's new code project shows an increase, but also more design base faults are identified compare to the average. Of course many parameters influence quality (e.g. the competence level of the project teams). A conclusion is that Agile project quality can still be controlled and the quality level remains high.

4.4 Hours per requirement

We also measured for the period of one year the cost per requirement

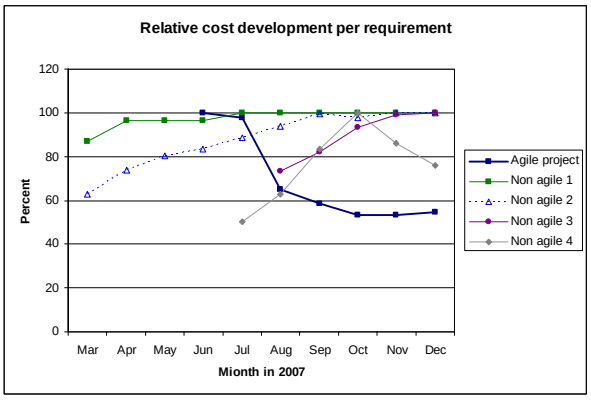


Fig. 2 Relative cost development per requirement

Figure 2 shows a chart of the relative cost development of an average requirement. For the Agile project the average cost declines after a couple of iterations. The team used about three iterations to become able to define user stories of about a constant amount of hours of implementation cost. This kind of learning is not visible in the non-Agile projects.

4.5 Employee Motivation

The pilots were initiated top-down, the pilot execution however, was bottom-up. Pilot projects received training and then had to find their own way forward with the help of an Agile coach. In general it took about three iterations before the Agile adepts got their feet back on the ground and Agile opponents got enthusiastic. At the end of the pilots people were motivated and engaged.

In one of the groups within the PDU a significant increase in accountability was observed. In this group (60 people) about half of the people participated in an Agile project. Accountability[2] and initiative for the whole group was rated positive by 71% of the people in May 2007. The amount of positive responses went up to 81% in October 2007. The total for the PDU was 71% in May and 73% in October. Best in class in Ericsson was 83% in October 2007. Learning[3] increased from 56% to 66% positive responses.

[2] Two sentences were requested to be rated (1 to 5, 4 & 5 being considered positive answers): "in my workgroup/team everyone accepts accountability for problems related to our work" and "my colleagues often act on their own initiative".

[3] Sentences to be rated for learning were: "I have the skills/qualifications I need to carry out my assignments at work", "At my work I get to share knowledge and experience with others regularly" and "I am familiar with how customers rate Ericsson as compared with our competitors".

4.6 Multi-disciplinary teams

It turned out that having all required disciplines in one team is highly effective. The TTM reduced because it became possible to do design and testing almost at the same time. This is an improvement compared to testing afterwards. In some cases the teams are doing the maintenance of products they delivered themselves, rather than a separate department. This all resulted in less handovers.
Besides that everything has become a team responsibility: developers are also responsible to testing activities and vise versa. Team members learn a lot about other disciplines and as a result the teams are more flexible; most people can pick up lots of different tasks, design or test.

5. Learning Points

5.1 The introduction

Starting overnight with Agile processes can be done without performance drawbacks on a project level. Very helpful preliminary steps were getting used to timeboxing instead of scope boxing and 'in-scoping'. This leaves the main focus of the change to shortening iterations and the "human/social" ways of working. The latter however requires constant attention and effort to make the change and make it stick.

5.2 Crossing unit boundaries

"Why change? Agile practices will not suit us." The further away from actual software development, the less understanding about what benefits Agile ways of working will bring. The idea that Agile works best if customer contact and customer contract are synergistic does not easily penetrate a large organization.

5.3 Inspiration sessions

Inspiration sessions are sessions in which external speakers share their experiences or teach how Agile or Lean can work. These sessions not only educate the Agile and Lean way of working, but also take away some fears. When people are more familiar with Agile their resistance reduces.

5.4 Agile testing

In terms of changes in their way of working we believe testers have had to adapt the most. Continuous integration, testers as members of the development teams, pairing with developers, no black box testing because testers knew much better what the developers were doing were all factors testers had to get used to. Additional ad hoc training was used to learn more about Agile testing [3].

5.5 Management support

In a bottom-up approach line and project management have limited involvement. However, line and project managers are a key in making the change stick, helping people resolve impediments and conflicts, building a learning organization and, above all, showing what is meant with Agile and lean ways of working. In literature

you find terms like ‘Quiet Leadership’ [4], ‘Light Touch Management’ [5] and ‘Builder of Learning Organizations’ [1]. This has to involve a mix of training and coaching for management.

5.6 Agile coach

It appeared highly valuable to have a coach support the Agile teams with their pursuit towards achieving a good mindset and towards achieving a way of working that continuously improved. The Agile coach recognized and sensed team needs that otherwise would have remained hidden. The Agile coach provided Agile training, set up of a network of retrospective facilitators and later on a network of internal Agile coaches. Having an experienced coach is a prerequisite for a change process towards Agile if experienced Agile developers and managers are lacking.

5.7 eXtreme Programming

In the beginning of the change process we mainly focused on implementing Scrum. Now, after more than one year of Agile experience, we start to apply the more technical XP practices as well. More teams are using continuous integration and are beginning to use Test-Driven Development (TDD). Developers started to use a test-first approach and now we are moving towards real TDD.

6. Conclusion

Agile practices have established a fragile and partial cultural change. Where change used to sound like “we have a new organization chart and we expect you to help establish the new goals”, we now had a change sounding like “we involve you all in improving our processes in order to improve our business and key performance indicators”. Oh, and “we have no new organization chart until we know how to change it for the better”. We are convinced that with good support and perseverance we can make this change stay [2].

7. References

[1] Implementing Lean Software Development: from concept to cash. M. and T. Poppendieck, 2007
[2] Our Iceberg is Melting. J. Kotter, H. Rathgeber, 2007
[3] Agile Testing, Anko Tijman
[4] Quiet Leadership, David Rock, 2006
[5] http://blog.cutter.com/2007/09/13/no-more-self-organizing-teams, introducing Light Touch Management, Jim Highsmith

Economische crisis: kostenbesparing vraagt om reviews

Het stormt buiten. De onheilstijdingen buitelen over elkaar heen en het lijkt erop dat een uitweg uit de algehele economische neergang voor onmogelijk wordt gehouden, zelfs door de grootste optimist. Toch is er reden voor optimisme want de historie laat zien dat bedrijven die zich richten op innovatie en kwaliteit de grootste kans hebben om weer boven te komen drijven. Dan moet er uiteraard nog wel een bedrijf zijn. Met andere woorden, de kosten dienen onder controle te zijn en kostenbesparing is een belangrijk aandachtspunt. De huidige crisis maakt de noodzaak om een product van voldoende kwaliteit tegen een verantwoorde prijs te leveren alleen maar duidelijker. Door het toepassen van reviews en inspecties, oftewel door ‘beter’ te testen aan de

voorkant, kunnen producten de eerste keer goed en goedkoper opgeleverd worden. Bij de huidige tegenwind verdienen reviews dan ook volop aandacht. De reviewtechniek heeft zich, mits goed ingevoerd, al vaak bewezen maar gaat het ons helpen om door deze crisis heen te komen?

Tijdens een review wordt een product onderworpen aan een beoordeling door iemand anders dan de auteur. Reviews leveren voordelen op in termen van tijd, geld en kwaliteit omdat de fouten die mensen maken bij het uitvoeren van hun taak vroegtijdig in het proces worden gevonden. Door middel van foutpreventie wordt bovendien voorkomen dat dezelfde fouten in de toekomst opnieuw worden gemaakt. De voordelen van deze techniek zijn al heel lang bekend en ook reeds bij vele bedrijven ingevoerd. Via een metrieken programma kan inzichtelijk gemaakt worden dat reviews leiden tot kostenbesparing en kwaliteitverbetering. Toch zien ook de auteurs dat het lastig is om onder de dagelijkse projectdruk, die door de crisis alleen maar toeneemt, vast te houden aan reviewtechnieken. Zonder in te willen gaan op de dieperliggende oorzaken hiervan (misschien is genezen wel leuker dan voorkomen en wellicht is ‘scoren’ met een noodoplossing meer motiverend dan het voorkomen van de noodsituatie, zoals het halen van de trein op het laatste moment vaak meer voldoening geeft dan eenvoudigweg op tijd van huis vertrekken) worden in dit artikel twee onderdelen van een reviewprogramma behandeld die van grote invloed zijn op het langdurig succesvol implementeren van reviews: modereren en metrieken.

De moderator bepaalt in hoge mate of reviews een succes worden en blijven. Hij of zij is niet alleen de leider van de loggingmeeting, maar van het gehele reviewproces. Gilb en Graham spreken over “a person trained to co-ordinate and control all peer review team activities” (Gilb e.a., 1993). Als leider van de review, vanaf de entrycheck tot aan de afronding, is de moderator verantwoordelijk voor het zo efficiënt en effectief mogelijk inzetten van de betrokkenen door onder andere het bepalen van rollen en referentiekaders. Hiermee wordt verkomen dat iedere deelnemer naar dezelfde fouten zoekt of dat deelnemers slechts vluchtig door een document gaan in de hoop zo de meest in het oog springende fouten eruit te halen. Door iedere deelnemer een andere reviewrol te geven en verschillende referentiekaders te definiëren kunnen deelnemers zich richten op een gedeelte van de fouten en worden er in totaal meer verschillende fouten gevonden in minder tijd: een kostenbesparing.

Er is nog een andere belangrijke reden om reviewrollen te verdelen. Reviews worden doorgaans als extra werk ervaren door de deelnemers. Met een overvolle takenlijst, mede doordat collega’s wegbezuinigd zijn, zullen deelnemers slechts bereid zijn iets extra’s te doen, mits de volgende voorwaarden goed zijn ingevuld:

- Het werk en de bijdrage van de deelnemers aan de review is duidelijk en overzichtelijk.
- Het reviewproces is goed georganiseerd.
- De uiteindelijke meeting wordt efficiënt en effectief geleid.

Dit alles vraagt om een duidelijke verdeling van reviewrollen en deelnemers. Hierin helpt een plan of reviewoverzicht. Het opstellen van een reviewoverzicht bestaat uit de volgende activiteiten:

1. bepalen welke producten worden gereviewd;
2. bepalen met welke reviewtypen de producten worden gereviewd;

3. bepalen wat het referentiekader is waartegen de producten worden gereviewd;
4. bepalen welk type deelnemers betrokken worden bij welke review.

Voor de projectleider en de moderator is zo'n overzicht de basis om te kunnen sturen op tijd, budget en kwaliteit tijdens de uitvoering van reviews.

Om zowel het gebruik als de resultaten van reviews te kunnen beoordelen, zijn metrieken essentieel. Metrieken zijn de instrumenten die aangeven hoe met reviews wordt gevaren. Zonder metrieken worden reviews al snel gezien als investering, omdat onduidelijk is wat de opbrengsten zijn. Metrieken kunnen in het kader van reviews voor meerdere doelen worden gebruikt zoals:

1. Het bepalen van de kosten als gevolg van lage kwaliteit, de vermijdbare faalkosten: de vermijdbare faalkosten zijn de kosten voor herstel van fouten die eerder in het ontwikkeltraject zijn gemaakt. Met de vermijdbare faalkosten wordt de beslissing om reviews te implementeren onderbouwd.
2. Het aantonen van de opbrengsten van reviews: op basis van het aantal gevonden fouten tijdens de review kan worden bepaald wat de potentiële besparing is. Door behalve het aantal fouten in het review proces ook het aantal fouten in het testproces te meten kunnen de opbrengsten concreet worden onderbouwd.
3. Het geven van informatie over de kwaliteit van het product dat gereviewd is: het aantal fouten dat bijvoorbeeld op een pagina wordt gevonden geeft informatie over de productkwaliteit. Bij een eerste meting zegt dit niets en is het vaak schrikken als ineens blijkt dat een document vijf fouten per pagina heeft. Dit kan echter worden verbeterd en een zichtbare verbetering van vijf naar bijvoorbeeld drie fouten stimuleert om verder te gaan.
4. Het kwantificeren van de kwaliteit van het ontwikkelproces, zo is een eenvoudige metriek het meten van de kwaliteit van het eindproduct in de vorm van bijvoorbeeld het aantal fouten per regel code of per functiepunt.

Metrieken geven inzicht en zeker metrieken die inzicht geven in problemen met de kwaliteit schrikken managers nogal eens af omdat ze niet hadden verwacht dat er nog 'zoveel' fouten in het product zitten. Het is dan ook zaak om metrieken te zien als een instrument dat inzicht geeft waar en wanneer er gestuurd moet worden en, misschien nog wel veel belangrijker: wat de opbrengsten zijn.

Bovenstaande en andere onderwerpen (o.a. de verschillende reviewtypen, implementatie, succesfactoren en een uniek groeimodel om de toegevoegde waarde van reviews te optimaliseren) benodigd om reviews succesvol in te voeren, worden behandeld in het boek "Reviews in de Praktijk, testen aan de voorkant". Het boek wordt gecompleteerd met een aantal cases uit de praktijk, direct bruikbare sjablonen en checklisten. Het boek is geschreven op basis van de review ervaringen van de vier auteurs, werkzaam bij drie verschillende bedrijven. De auteurs zijn: Jan Jaap Cannegieter, adjunct directeur en consultant bij SYSQA B.V., Erik van Veenendaal, directeur en managementconsultant bij Improve Quality Services B.V., Eric van der Vliet, software architect bij Logica Nederland B.V. en Mark van der Zwan, senior consultant en docent bij Improve Quality Services B.V.

De ervaringen zijn opgedaan tijdens de soms weerbarstige praktijk bij het invoeren van reviewprogramma's in zeer diverse organisaties. Dit heeft geleid tot een "zeer op

de praktijk gericht" boek waarover Meile Posthuma in Testnet Nieuws van december 2008 verder zegt: "Naast het boek van Tom Gilb en Dorothy Graham: Software Inspections, mag dit boek zeker niet ontbreken." In de huidige storm biedt het efficiënt inzetten van de zeer praktische review techniek nog meer kansen, mits de juiste instrumenten worden gebruikt om ons door deze crisis heen te navigeren.

Februari 2009
Eric van der Vliet, eric.van.der.vliet@logica.com
Mark van der Zwan, mzw@improveqs.nl

Software Testen werkt niet!

Inleiding

Uit een recent onderzoek in Noord Europa blijkt dat 89% van alle respondenten problemen heeft met systemen binnen 48 uur na de oplevering (bron: Logica). Falende software-projecten, stellen hoogleraren informatica Chris Verhoef en Jan-Friso Groote in de Automatiseringsgids, zorgen wereldwijd jaarlijks voor maar liefst 290 miljard dollar schade. Het tweetal is behoorlijk pessimistisch over de kwaliteit van software: 'Slechts 20 procent van de softwareprojecten slaagt, 30 procent eindigt in totale mislukking, en de helft voldoet niet aan de verwachtingen.'

Zorgelijke cijfers. Aandacht voor de kwaliteit bij de ontwikkeling van nieuwe software systemen is dan ook van groot belang voor ieder bedrijf. Zoals uit dit artikel zal blijken is het testen van software voor de ingebruikname daarbij een onvoldoende waarborg.

Software testen

Opdrachtgevers gebruiken voor kwaliteitscontrole vooral Functional Acceptance Tests en een System Integration Test. Bij deze twee manieren van testen, laat men de software draaien, al dan niet in een testopstelling, en al dan niet gekoppeld met alle andere systemen en databases. Het principe van beide methodes is hetzelfde: kijken of het programma doet wat het moet doen.

Het lastige van testen is dat men maar een deel van de kwaliteitsaspecten onder de loep kan nemen. Om te bepalen in welke mate het testen van software voldoet als kwaliteitswaarborg nemen we vier IT-audit invalshoeken:

1. Effectiviteit: de mate waarin geplande activiteiten en geplande resultaten werkelijkheid worden. Software is pas effectief als het onderhoud goed te verzorgen is. Als de softwareontwikkelaar zijn werk effectief heeft gedaan is de software onderhoudbaar en optimaal herbruikbaar.
2. Efficiency: de verhouding tussen behaalde resultaten en de gebruikte middelen. Programma's kunnen flink verschillen in hun mate van zuinigheid, kan een programma in 20 minuten gedraaid worden, of is daar 24 uur voor nodig?
3. Betrouwbaarheid: de mate waarin gegevens in overeenstemming met de feiten zijn. Klopt wat is opgeslagen in de computer met de buitenwereld? Is het systeem betrouwbaar, dan steunt het op twee aspecten van softwarekwaliteiten: juistheid en volledigheid. Dat is te testen; als er iets wordt ingetikt en bewerkt, komt er dan uit wat er was verwacht?

4. Continuïteit: de mate waarin gegevensverwerking ongestoord voortgang kan vinden. Dat heeft te maken met: bedrijfszekerheid, repareerbaarheid (softwarefout moet snel gerepareerd kunnen worden), robuustheid en herstelbaarheid (de gegevensverwerking moet weer snel hersteld kunnen worden na een crash).

Wat direct opvalt, is dat testen voor een aantal onderdelen ontoereikend is. Onderhoudbaarheid en herbruikbaarheid zijn bijvoorbeeld niet te testen. Onderhoud en hergebruik gaan over de manier waarop iets is geschreven of opgezet, het gaat om stijl, het gaat om structuur, het gaat om consistentie, het gaat om good-programming-practice. Als die niet toegepast zijn, zal een test daar niets van blootleggen omdat alleen naar buitenkant wordt gekeken. In auditterminologie; er wordt gekeken naar bestaan en werking van systemen, zonder dat de opzet wordt onderzocht.

Ook de mate van repareerbaarheid, valt met testen niet waar te nemen. Repareerbaarheid is zeer belangrijk voor de continuïteit van bedrijven. Repareerbaarheid is een ondergeschoven kindje: het is belangrijk om software effectief en doeltreffend te kunnen repareren, en dan nog het liefst zo snel mogelijk, in ieder geval binnen een voorspelbare tijd; binnen een uur, binnen vier uur. Dat betekent óók, niet onbelangrijk, dat een reparatie niet méér fouten introduceert dan dat ze oplost. Eigenlijk is repareerbaarheid nog belangrijker dan onderhoudbaarheid, omdat een programmeur voor een reparatie per definitie minder tijd tot zijn beschikking heeft. In het geval van onderhoud kunnen altijd nog meer geld en tijd uitgetrokken worden, iets wat bij reparaties nauwelijks extra succeskansen oplevert.

Voor zuinigheid, juistheid, volledigheid en bedrijfszekerheid geldt dat ze allemaal deels te onderzoeken zijn met Functional Acceptance Tests en een System Integration Test. Met testen is het mogelijk om een route door de software nemen, waarin er niets fout gaat, maar dat zegt nog niet zo veel. Er kunnen dingen gebeuren die maar eens per jaar voorkomen; bijvoorbeeld het tegelijk vollopen van een schijf en het uitvallen van de communicatie. Als die twee tegelijk gebeuren, kan het systeem plat gaat, terwijl het systeem nooit is getest onder die omstandigheden. Het kan zijn dat het programma doorgaat onder die omstandigheden, terwijl het zou moeten stoppen. Met functioneel testen wordt dat probleem niet blootgelegd en met integratietesten is dat duur en tijdrovend.

Software inspecteren

Zoals blijkt uit het voorgaand helpt software-testen bij het vinden van fouten, maar een waarborg voor kwaliteit is het niet. De kwaliteit van bijvoorbeeld een nieuwe keuken kan niet worden vastgesteld door een ei te bakken. Om verborgen gebreken op te sporen is een inhoudelijke inspectie noodzakelijk. Hierbij worden fouten blootgelegd die bij testen gewoonweg niet gesignaleerd worden.

Hoeveel fouten zitten er gemiddeld in de source-code? De schatting van Frits Vaandrager, hoogleraar Informatica van Technische toepassingen aan de Radboud Universiteit Nijmegen: gemiddeld één fout in elke paar honderd regels software. ‘En van die fouten is aantal procent zo ernstig dat ze een systeem kunnen laten stilvallen.’ (Bron: NRC Handelsblad).

Bij Software Inspectie wordt de kwaliteit van software beoordeeld door te kijken naar de source-code zelf. Een programmeur is gemiddeld 48 uur bezig met één A4 aan source-code. De “review” hiervan door een software inspecteur kost maar enkele minuten. Gezien het bedrijfsbelang van software kwaliteit en de zelfstandigheid waarmee programmeurs werken zou deze kwaliteitscontrole een minimale vereiste moeten zijn.

Bij een Software Inspectie wordt met een rode pen in de uitgeprinte source-code gezocht naar afwijkingen van kwaliteitsnormen die te maken met betrouwbaarheid en onderhoudbaarheid.

Betrouwbaarheid

De betrouwbaarheid van software wordt bepaald door de mate waarin de programmatuur goed functioneert onder bijzondere omstandigheden, zoals onlogische invoer, stroomuitval en andere onvoorziene situaties. Inspectie op onderstaande kwaliteitscriteria geeft inzicht in de betrouwbaarheid van software.

Ready for release (RR); Bij de in gebruik name van de software mag er geen code meer aanwezig zijn die uitsluitend voor software-ontwikkelingsdoeleinden is toegevoegd. Programmeurs schrijven bijvoorbeeld vaak “tijdelijke” code om de software beter of sneller te kunnen testen. Soms blijft deze code onbedoeld in de productiecode staan.

Transaction consistency (TC); De wijze waarop het systeem gegevens verwerkt en bewaard moet volledig consistent zijn. Waar men bijvoorbeeld de performance van een systeem verbetert, ontstaan vaak gaten in deze consistentie. Gaten die lang onopgemerkt kunnen blijven bestaan totdat bepaalde interacties zich voordoen. De eis van Transaction Consistency stelt dan ook hoge eisen aan de programmeurs, vooral waar meerdere gebruikers simultaan aan dezelfde gegevens werken.

Error handling (EH); De error handling moet compleet en consequent zijn. Indien het programma onverhoopt faalt, moet de error handling ervoor zorgen dat de fout op een consistente wijze wordt afgehandeld. Daarnaast moet de error handling inzicht geven in de oorzaak.

Defensive programming (DP); Het systeem moet dusdanig geprogrammeerd worden dat het stopt indien iets onvoorziens gebeurt. Het systeem mag bijvoorbeeld niet in een loop raken, vastlopen of onlogische invoer accepteren. Ter voorkoming van schadelijke gevolgen bij een onvoorziene situatie dient het systeem zichzelf correct af te sluiten en de gebruiker te voorzien van een foutmelding.

Onderhoudbaarheid

De onderhoudbaarheid van software wordt bepaald door de mate waarin aanpassingen efficiënt gerealiseerd kunnen worden. Door inspectie op onderstaande criteria kan worden voorkomen dat een systeem in gebruik wordt genomen waarvan gerepareerde reparaties steeds opnieuw gerepareerd zullen moeten worden!

Structured programming (SP); In een goed opgesteld programma is voor alles een plaats en staat alles op zijn plaats. Dit betekent bijvoorbeeld dat de code geen onnodige herhalingen bevat. Hiermee wordt gewaarborgd dat een gemiddeld stuk programmatuur door een gemiddelde programmeur veilig kan worden onderhouden.

Consistent programming (CP); De namen en structuur van modules met vergelijkbare functionaliteit moeten onderling consistent zijn. Bij een software inspectie wordt daarom naar inconsistentie in de softwarecode gezocht. Inconsistentie kan duiden op een algemeen kwaliteitsprobleem van de module waarin het is gevonden, of zelfs van het systeem als geheel.

Consistent layout (CL); Analoog aan de wereld van “normale” documenten is de consistente lay-out van programmatuur een hulpmiddel om de structuur ervan inzichtelijk te maken. Een inconsistente lay-out is misleidend en werkt verhinderend bij onderhoudswerkzaamheden.

Isolation of parameters (IP); Bij het schrijven van de source-code kan het lijken alsof teksten en getallen een vaste waarde hebben. Mocht de situatie echter wijzigen, dan moeten aanpassingen efficiënt gerealiseerd kunnen worden. Dit kan door de vaste waardes als parameters te isoleren.

Aan de hand van onderstaande voorbeelden worden grote consequenties van kleine afwijkingen in de kwaliteit van software inzichtelijk gemaakt.

Voorbeeld 1: Energiebedrijf slaat flater

De automatische incasso van een energiebedrijf werkte niet goed. Bij tienduizenden klanten werd het geld voor één maand energie niet automatisch afgeschreven.

Bovengenoemde situatie is ontstaan na ingebruikname van een nieuw incassoprogramma. Zes maanden nadat het programma in gebruik genomen is, loopt op een avond de disk vol. De laatste 20.000 van de 800.000 incasso’s worden daardoor niet weggeschreven. Het programma gaat vrolijk door.

De consequenties zijn aanzienlijk. Een kleine drie miljoen euro omzet wordt niet geïnd. Drie maanden later moet een deurwaarder ingezet worden om de laatste 70.000 euro ervan te innen. Het bedrijf heeft het vertrouwen van tienduizenden klanten geschaad.

Oorzaak: Een ontbrekende regel programmatuur.

Voorbeeld 2: Toonaangevende uitgever publiceert verkeerde informatie

Bij een uitgever van financiële informatie wordt voor de derde keer in één maand de redactiedeadline niet gehaald. De oorzaak ligt bij de schermen die gebruikt worden om bepaalde gegevens handmatig bij te werken. De schermen blijven lang “hangen”. Uit onderzoek op de IT-afdeling blijkt dat de schermen geblokkeerd worden door een batchproces. De programmeurs krijgen de opdracht de bottleneck op te lossen en passen daarom de betreffende programma’s aan.

Drie maanden later belt een abonnee met de mededeling dat een gepubliceerde koerswaarde niet klopte. Naar aanleiding van een flinke interne ruzie tussen de IT-afdeling en de redactie wordt onderzoek verricht. Daaruit blijkt dat in de afgelopen week alleen al vijf fouten zijn gepubliceerd. De fout is telkens van dezelfde aard; het lijkt erop dat de koers niet bijgewerkt is. Het hoofd van de IT-afdeling besluit om er een expert bij te halen.

Oorzaak: Bij de onderhoudswerkzaamheden is de consistentie van het programma beschadigd.

Men besluit een audit van de software te laten verrichten. Tijdens zijn inspectie legt de auditor tientallen ernstige programmeerfouten bloot.

Return on investment

IBM heeft vastgesteld dat ieder uur besteedt aan software inspectie 102 uur oplevert; 20 uur testen en 82 uur herstelwerk aan de software. (Bron: Holland, ‘Sofware Inspection as an Instrument of Change’, Software Quality Professional). Bij ICI (Imperial Chemical Industries) hebben ze ontdekt dat de onderhoudskosten voor ongeïnspecteerde programma’s tien keer hoger liggen geïnspecteerde software. (Bron: Gilb and Graham, Software Inspection).

Door in een vroeg stadium fouten te ontdekken en te corrigeren, kunnen problemen in volgende stadia van de ontwikkeling worden voorkomen. Een bedrijf dat Software Inspectie bij maatwerk software toepast, profiteert van een betere voorspelbaarheid van zowel de kosten als de softwareprestaties, lagere kosten voor ontwikkeling en onderhoud, verhoogde klantentevredenheid, en een beter moreel onder de programmeurs.

Het rendement op Software Inspectie ontstaat doordat hogere kosten in de toekomst worden voorkomen. De kosten om een ernstig tekort te ontdekken en te verbeteren in een vroegtijdig stadium zijn altijd vele malen lager. Men zegt niet voor niets: voorkomen is beter dan genezen.

Inspectie als integraal onderdeel van softwareontwikkeling

Software Inspectie is het meest effectief wanneer u het een integraal onderdeel maakt van uw softwareontwikkelingstraject. Door Software Inspectie te combineren met workshops waarin met programmeurs aandacht wordt besteed aan ontwikkelingspunten zal niet alleen de kwaliteit van de ontwikkeling maar ook die van uw programmeurs verbeteren. Van zo’n workshop gaat een verfrissende werking uit, omdat in de drukke praktijk van alledag weinig tijd is voor zelfreflectie. Door middel van voortgangsgesprekken en een finale Software Inspectie haalt u het beste uit uw mensen boven. En daar gaat het om, want softwareontwikkeling is immers mensenwerk.

Zelf doen of uitbesteden?

Kwaliteitscontrole is een vak apart. Veel bedrijven betrekken dan ook nu al een strategische partner bij de kwaliteitscontrole op softwareontwikkeling. Het eerder genoemde onderzoek geeft de volgende redenen om een strategische partner te betrekken bij de kwaliteitscontrole:

- 41% van de respondenten noemt betere kwaliteit van software;
- 38% van de respondenten noemt betere waarborg op het ontwikkelingsproces;
- 37% van de respondenten noemt verhoging van de “accountability”, en;
- 36% van de respondenten werkt met een strategische partner om een snellere time-to-market te garanderen.

Graham Bolton, Director
Institute for Software Quality (www.ifsq.org)

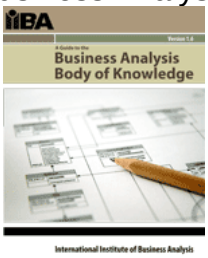
Hoera, een nieuw boek over requirements

Succes met de requirements, een praktische toelichting bij een praktisch boek

Onder de titel ‘*Succes met de requirements!’* is recentelijk bij Academic Service een nieuw Nederlandstalig boek verschenen over Requirements (auteurs: Arendsen, Cannegieter, Grund, Heck, de Klerk en Zandhuis).

Waarom een boek over requirements?

Je zou je af kunnen vragen waarom er opnieuw een boek over requirements is verschenen. Is er al niet genoeg over dit onderwerp geschreven? En in het verlengde daarvan, wat kan je hiermee in de praktijk?
Nu wil het geval dat ik zelf meer dan gemiddeld geïnteresseerd ben in Requirements. Het is een gebied waar ik in de dagelijkse adviespraktijk telkens mee te maken heb. Verder onderhoud ik vanuit het SPIder bestuur contacten met de WG Requirements. Dat is dus beslist geen toeval! Ook heb ik recentelijk in een groot compliancyprogramma bij een bank een end-to-end Requirements Engineering proces ontwikkeld en ingevoerd. Dus vanaf Business Needs en Requirements naar de IT provider en weer terug tot en met acceptatie en vrijgave in productie. Waar ik normaliter mijn Volere, Wiegers en Gilb boeken als bron gebruik om iets met RE te doen, heb ik besloten om deze keer dit Requirements boek als leidraad te kiezen. Na eenmaal het boek vluchtig te hebben gelezen sprak de opzet me aan, vandaar. Ongeveer in dezelfde periode werd ik getipt op de IIBA, International Institute of Business Analysis – Guide to the Business Analysis Body of Knowledge (ref. 1).



In de momenteel beschikbare Release 1.6 Draft van de IIBA ontbrak nog de nodige tekst, maar ik was wel onder de indruk van de compleetheid van deze standaard. Ik

raad dan ook iedereen aan die op het gebied van Requirements bezig is, de IIIBA tot je te nemen. Op 31 maart 2009 wordt een nieuwe versie van de IIBA verwacht.
In de rest van dit artikel ga ik in op bepaalde onderdelen uit het boek *Succes met de Requirements!*, vertel ik waar ik ben afgeweken en stel ik wat zaken ter discussie.

Over de launch van het boek

In 2008 werd het nieuwe boek '*Succes met de requirements!"* in een tweetal sessies via presentaties in zowel Eindhoven en in Amstelveen aan het publiek voorgesteld. Zelf was ik op de bijeenkomst bij Logica in Amstelveen aanwezig en merkte dat er best een grote opkomst en dus belangstelling voor Requirements blijkt te bestaan. Al pratend met de inleiders bleek dat ook in Eindhoven het geval geweest te zijn. Daar werd het boek op de TU bij LaQuSo gepresenteerd. Ook daar was sprake van een prima opkomst. Wat is dan het geheim achter dit succes vroeg ik me af. Komt het omdat er maar erg weinig Nederlandstalige boeken over Requirements verkrijgbaar zijn of is er een andere reden voor?
Uit de presentatie van Arno Grund selecteerde ik wat stellingen (in vet) en voegde daar wat commentaar van mezelf aan toe.

Stelling 1. **De requirementsanalist is een kenner.** Tegen deze stelling zal op zich geen bezwaar zijn, maar om welke kennis gaat het nu eigenlijk? Betreft het businesskennis, functionele kennis of juist technische kennis? Betreft het alleen kennis van de inhoud, de context, of ook/juist proceskennis. Kortom deze stelling is duidelijk niet SMART geformuleerd. Requirementsanalisten werken in het Businessdomein, maar ook in functionele en technische domeinen. Elk domein veronderstelt andere kennis. Niet al die kennis zal altijd bij één persoon beschikbaar zijn. Sterker nog (met name in grotere organisaties) zijn Requirementsanalisten gespecialiseerd en zullen indringend moeten samenwerken en hun kennis moeten bundelen om de juiste en volledige Requirements boven water te krijgen.

Stelling 2. **Materiedeskundigen hebben altijd gelijk.** Deze stelling herkennen we ook denk ik zo. Maar wie controleert nu eigenlijk de Materiedeskundigen? En wat als ze toch ongelijk hebben? Wat je ook doet, in de praktijk kan je dit nooit voor 100% dichttimmeren. Het antwoord zit 'm in verantwoordelijkheidstelling. Voor een Businessproject (programma) kan alleen de Business verantwoordelijk zijn. IT kan en zal meehelpen, kan soms tegengas geven (tegen te luxe wensen) en zal op consequenties ten aanzien van onderhoudbaarheid, risico's en Total Cost of Ownership wijzen, maar de Business beslist. Als in deze stelling met materiedeskundigen de Business wordt bedoeld zou de stelling best wel eens waar kunnen zijn.

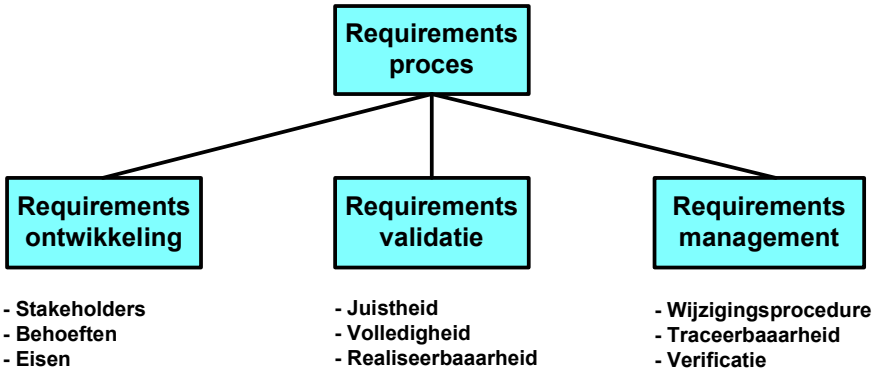
Stelling 3. **De requirementsanalist is een zeurpiet.** Ook deze stelling herkennen we ook wel denk ik zo. Waarom? En hoe zit dit dan? En wat doe je in dat geval? En wanneer treedt dit op? Vragen, vragen, doorvragen en nog eens vragen. Het hoort er gewoon bij. Van vragen wordt je wijzer is een bekend gezegde. Als Requirementsanalist moet je een zeurpiet durven zijn, een advocaat van de duivel. Je doet dit om het werkelijke probleem goed te kunnen begrijpen en het probleem achter het probleem te kunnen ontdekken om daarmee de best mogelijke oplossing voor de Business te verkrijgen tegen een aanvaardbare prijs.

Stelling 4. **Managers zijn ook gebruikers.** Die stelling deel ik van harte. Een van de eerste en belangrijkste dingen die je moet doen als Requirementsanalist is het identificeren en rubriceren van Stakeholders. *Succes met de Requirements!* onderschrijft dit. Hierbij hoort ook het maken van een inschatting van het belang van een individuele Stakeholder en van de krachtenverhoudingen tussen die Stakeholders. Door hun positie ben je misschien geneigd eerder naar een manager te luisteren dan iemand die operationeel werkend is, maar het zal niet de eerste keer zijn dat een manager een grote broek aantrekt maar feitelijk van toeten nog blazen weet.

Stelling 8. **Een datamodel is ook maar een requirement.** Deze stelling vind ik persoonlijk een hele gevaarlijke. Ik heb al jaren begrepen dat er grofweg in RE land twee kampen zijn. Mensen die van mening zijn dat een Use Case, een datamodel of een Class model een Requirement zijn en mensen die vinden dat deze modellen weliswaar nuttig en noodzakelijk zijn, maar alleen de totstandkoming en validatie van Requirements ondersteunen en zelf absoluut geen Requirements zijn. Zo ben ik het ermee eens dat een discussie over optionaliteit en kardinaliteit van entiteiten in een datamodel tot aanscherping en beter inzicht in Requirements leidt, maar ik vind niet dat deze modellen in de plaats treden van Requirements. Ze spelen wel een belangrijke rol in de validatie van Requirements.

Omgaan met Niet Functionele Requirements was de titel van de presentatie van Serge de Klerk. Hij doet een aantal uitspraken waar ik best achter kan staan. Echter in de schema's van bijlage C uit *Succes met de Requirements!* wordt het niet echt duidelijk gemaakt wat je vastlegt over Non Functionals en hoe je dit vastlegt. Figuur C.2 geeft wel een proces 1.5c 'Specificeer kwaliteitseigenschappen' weer, maar heeft geen eigen vastlegging anders dan dat resultaten aan Concept Use case specificatie en Concept SRS worden toegewezen. Ik ben op zich niet blij dat de auteurs van het boek mede leunen op Use Cases (zonder expliciet stelling te nemen of Use Cases nu wel of niet gelijk gesteld moeten worden met Requirements), maar in dit geval had ik me goed kunnen voorstellen als Supplementary Specification als vastlegging van Non Functionals zou zijn gebruikt.

Het belang van Requirements. Uit de presentatie van Jan Jaap Cannegieter over het belang van Requirements selecteerde ik het volgende overzicht:



Requirementsvalidatie valt in mijn visie op een professioneel RE proces uiteen in **Verificatie** en **Validatie**. Verificatie betreft de vraag of de Requirements wel goed zijn gespecificeerd (voldoet een Requirement aan de standaard specificatie van een Requirement, resp. Requirements template). Conform de Quality Gate van Volere

betreft dit een kwaliteitscheck op één individuele Requirement. Het resultaat is Passed of Failed. Validatie daarentegen betreft de vraag of wel de juiste Requirements zijn geselecteerd en of de Requirementsset wel een adequate oplossing biedt voor het Businessprobleem en realisatie daarvan ook nog eens past in de Project Constrants, zoals beschikbaar budget en doorlooptijd. Validatie is duidelijk vanuit Businessperspectief, uitgevoerd door Business Stakeholders en dus een duidelijke Businessverantwoordelijkheid. Naast Verificatie en Validatie heb je nog het fenomeen Work Order (project). Pakketten van door Stakeholders gevalideerde Requirements worden (doorgaans door de Requirements Manager) samengesteld zodat deze aan Project Stakeholders kunnen worden aangeleverd en in een project kunnen worden gerealiseerd. De projectmanager in kwestie representeert in deze de Project Stakeholders en zegt 'ja ik wil en ik kan' tegen de Business. Dit wordt vaak via een formele approval door een project stuurgroep bekrachtigd.

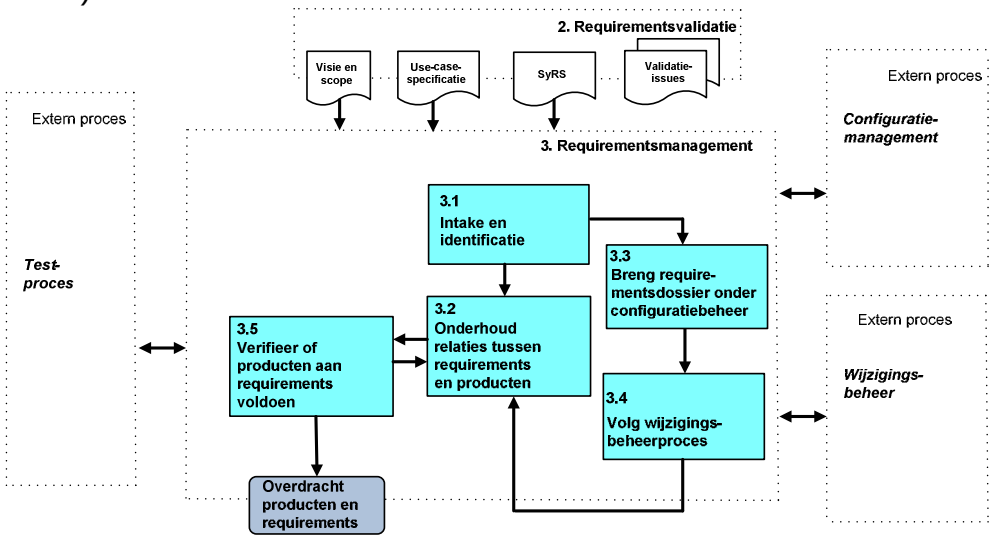
Praktische ervaringen en hindernissen

Bij de uitwerking van een RE proces moet goed aandacht worden gegeven aan het onderscheid van de taken en verantwoordelijkheden van de Requirements Manager versus de 'Requirements Acquirer', de Business analist, Requirements Engineer, System Matter Expert, informatieanalist of onder welke naam een persoon ook werkt in Requirements Elicitation, Analysis en Specification. Een valkuil is dat de Requirements Manager te veel werk (en verantwoordelijkheid) naar zich toetrekt en dat de Requirements Acquirer dit eigenlijk wel best vindt.
Er zijn vele verschillende ontwikkeling aanpakken, respectievelijk methoden & technieken en idem vele typen organisaties die een RE proces toepassen. De naamgeving van de gehanteerde Requirements types verschilt nogal. *In Succes met de requirements!* is gekozen voor Business Requirements, Gebruikers Requirements en Systeem Requirements (paragraaf 2.3). Deze logische en goed herkenbare drietrapsraket kom je vaak tegen in Requirements land. Immers aan opdrachtgever kant (Business) spelen hoog niveau, probleemgerichte Requirements, aan (IT) opdrachtnemer kant spelen laag niveau, dicht tegen technische realisatie gerichte Requirements. De Functionele Requirements, ook wel User Requirements of Product Requirements genoemd, specificeren wat gewijzigd moet worden aan informatiesystemen, respectievelijk applicatiesystemen die de Business processen ondersteunen. Dit is een logisch Requirements tussenniveau.
De toelichting op de verschillende Requirements types had van mij een tikje concreter mogen zijn (paragraaf 2.3). Ik heb op zich geen commentaar op de definitie van Business Requirements. Ik ben blij met de toevoeging van de definitie van Gebruikers Requirements dat de term 'gebruikers' breed gedefinieerd moet worden (primaire Business gebruikers maar ook onderhoud en beheer, exploitant, controle-instanties etc. gebruikers). Ik vind Systeem Requirements minder goed gedefinieerd. "Systeem Requirements zijn eisen of beperkingen waaraan het systeem dient te voldoen." Op deze manier weergegeven lijken Systeem Requirements zelf geen Requirements te zijn. Constraints zijn al van toepassing op een hoger Requirements niveau en natuurlijk ook op System Requirements niveau, maar er is méér dan Constraints. Zelf zie ik Systeem Requirements als het laagste niveau van Requirements die uiteindelijk de input vormen voor het creëren van Functioneel Detail Ontwerpen (in SDM termen gedefinieerd) waarbij Systeem Requirements expliciet worden toegewezen aan afzonderlijke subsystemen.

"ICT is primair verantwoordelijk dat het systeem dat wordt gerealiseerd voldoet aan de Requirements. ICT is dus primair verantwoordelijk voor requirements-management" (paragraaf 2.5). Zoals het hier staat lijkt het er op dat Requirements Management niet aan de orde is aan de opdrachtgever/Businesszijde. Dit lijkt me niet correct. Juist aan opdrachtgeverzijde ligt de verantwoordelijkheid dat goede en volledige Requirements worden opgebouwd en worden beheerd/gemanaged. De oplevering van een systeem/product als resultaat van gerealiseerde Requirements wordt dan ook getoetst tegen de Requirements van de opdrachtgever. In het programma waaraan ik heb gewerkt, was de centrale Requirements Manager de persoon bij uitstek die namens de Business kan managen of de provider wel de goede producten oplevert en ook binnen de daarvoor afgesproken toleranties, zoals tijd en geld.
Bij de bespreking van analysetechnieken wordt gesteld dat de requirementsanalist Requirements die eigenlijk geen eis of beperking zijn te verwijderen (paragraaf 4.2). Ik ben voorstander van het principe om eenmaal vastgelegde Requirements nooit meer te verwijderen en zeker niet te verwijderen als er al een verificatieslag overheen is gegaan! De betreffende Requirement kan beter via een status als ingetrokken worden aangemerkt.

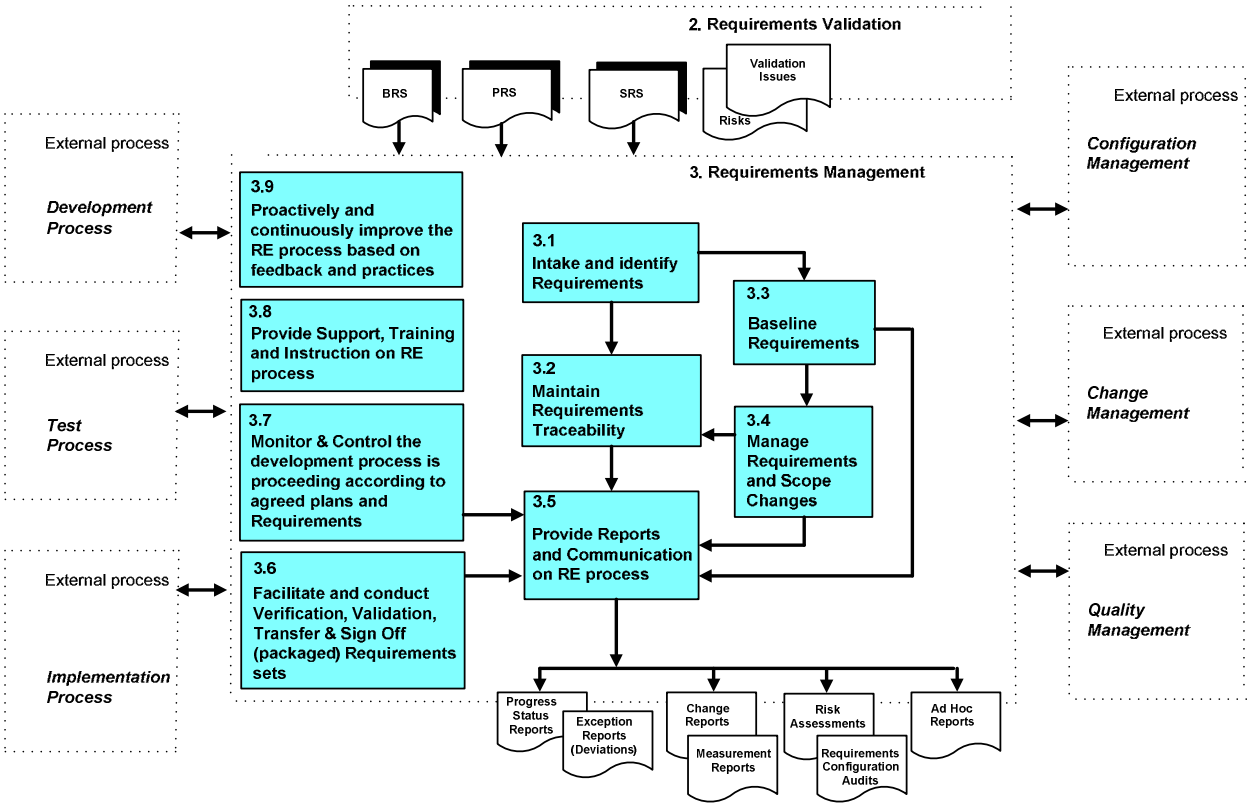
Requirements Management

Onderstaande figuur is ontleend aan het boek *Succes met de requirements!* (bijlage C, figuur C.4)



Succes met de requirements!
Figuur C.4: Proces requirementsmanagement

Ter vergelijking is hieronder het Requirements Management proces opgenomen zoals in het genoemde programma uitgewerkt.

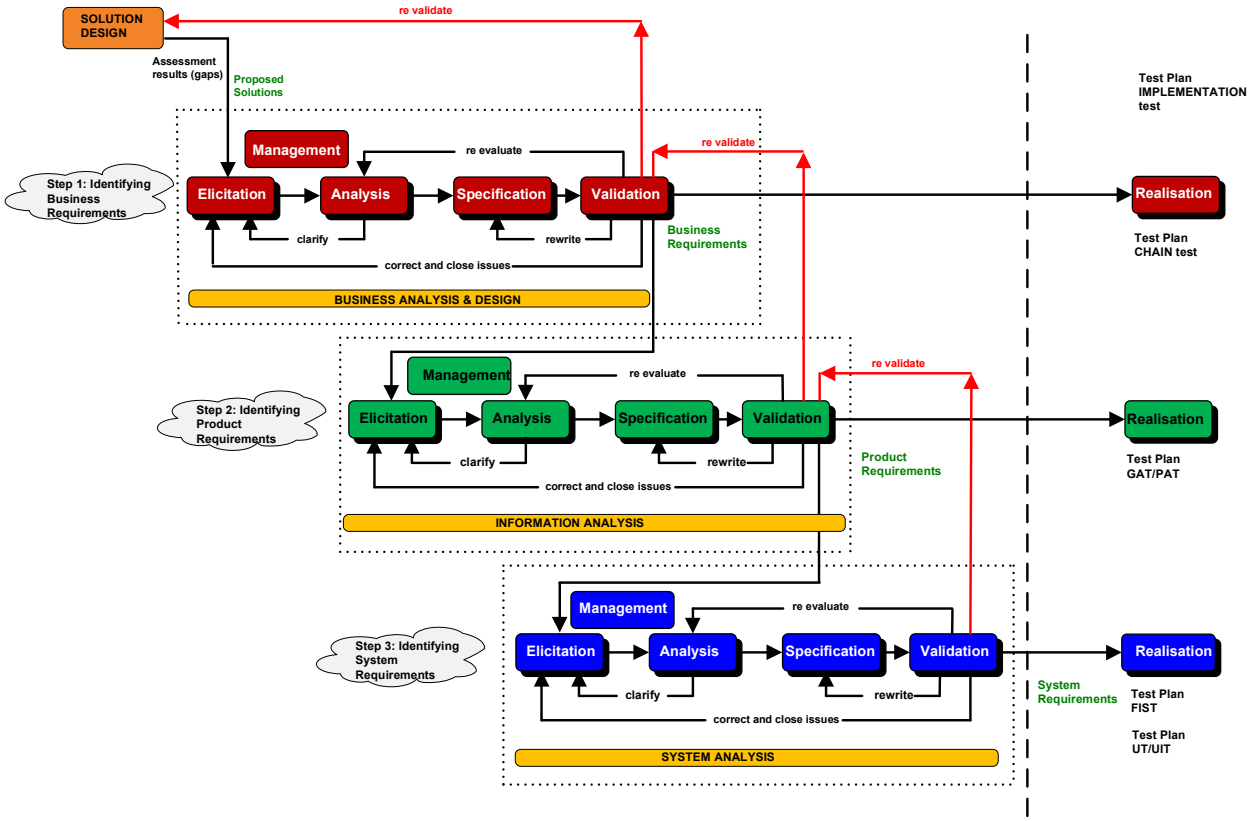


Succes met de requirements!
Figuur C.4: Proces requirementsmanagement

Merk op dat er concreet een subproces is gedefinieerd voor transfer en Sign Off van (packaged) Requirements sets. Ook een continu verbeterproces heeft hier een plaats in gekregen.

Het iteratieve en incrementele karakter van Requirements

Het RE proces is in uitvoering complex. Elicitation, Analysis en Specification worden in een Requirements Development proces gerealiseerd. Daarnaast worden Requirements gevalideerd (Verificatie en Validatie). Dit Requirements Development proces kent verschillende feedback naar vorige processtappen. Omdat het RE proces deels parallel/overlappend wordt uitgevoerd voor Business Requirements, Product Requirements en Systeem Requirements zijn tussen de drie Requirements abstractielagen eveneens verschillende feedback loops van toepassing. In een aantal slagen worden Requirements ontdekt, bekeken, getoetst en geleidelijk aan verfijnd en gedecomponeerd naar concretere SMART Requirements. Het eindresultaat van het proces is voldoende helderheid over WAT het project precies moet opleveren om met vertrouwen de ontwerpfase (en vervolgfasen) in te gaan. In onderstaande figuur zijn de drie Requirements abstractielagen in verschillende kleuren weergegeven.



Tot slot

Ik ben van mening dat we heel blij moeten zijn met het boek *Succes met de requirements!* Ik geef hierbij dan ook mijn welgemeende complimenten aan de auteurs! Het is makkelijk om aan de zijlijn commentaar en kritiek te geven maar ik weet als geen ander hoe lastig het is om een boek zoals *Succes met de requirements!* te publiceren. Je kunt je de vraag stellen of er nog meer behoefte in de markt zou zijn om praktische, vlot leesbare en in het Nederlands geschreven boeken over het onderwerp Requirements te ontwikkelen. Zelf ben ik daar zeker zijn. Na het verschijnen van het uitstekende boek 'begin bij het eind' lijkt in Nederland een trend te zijn ingezet om Requirements Engineering voor een breed geïnteresseerd publiek beschikbaar te stellen.

Vanuit het SPIder bestuur kunnen we alleen maar dit soort van ontwikkelingen stimuleren en van harte ondersteunen. Wie schrijft het volgende Requirements boek? Mogelijk dat je via SPIder professionele ondersteuning kunt krijgen om hiervan een succes te maken.

Martin Muller
SPIder bestuur

Martin.Muller@Logica.com

Referenties:

1. International Institute of Business Analysis – Guide to the Business Analysis Body of Knowledge
http://www.theiiba.org/AM/Template.cfm?Section=Body_of_Knowledge

Ervaring van een cursist: IREB Requirements Certificatie

Wat is IREB?

Binnen het vakgebied Requirements Engineering is een aantal jaren geleden de International Requirements Engineering Board (IREB) opgericht (www.certified-re.de). IREB is een internationale non-profit organisatie die zich richt op het verdere professionaliseren van het vakgebied. (Vergelijkbaar aan ISTQB op het gebied van testen.) Binnen IREB zijn onder andere requirements engineering experts zoals Chris Rupp en Suzanne Robertson actief. IREB heeft een certificatie-programma ontwikkeld voor Requirements Engineering, onderverdeeld in een drietal niveaus: Foundation, Advanced en Expert. Improve Quality Services is in Nederland geaccrediteerd tot het verzorgen van de cursus “Certified Professional for Requirements Engineering Foundation level” die volledig voldoet aan de eisen, zoals deze door IREB zijn vastgesteld. De cursus behandelt zowel de internationale terminologie als standaarden, technieken en methoden ten aanzien van Requirements Engineering en Requirements Management.

Persoonlijke motivatie

Als doelgroep voor het volgen van een opleiding- en certificeringtraject zoals IREB denk je waarschijnlijk primair aan mensen uit het vakgebied waar die requirements daadwerkelijk opgesteld worden: de Informatie Analisten, Business Analisten, etc. Maar waarom zou deze training ook niet zeer geschikt kunnen zijn voor mensen uit de discipline die controleert in hoeverre die requirements ook correct opgesteld en geïmplementeerd zijn: het testen? Zelf vind ik in ieder geval dat ik uitstekend tot de doelgroep gerekend kan worden; vandaar dat ik deze IREB-training dan ook ben gaan volgen. Als testprofessional ben ik al ruim tien jaar actief in het testvak en merk ik als externe medewerker bij diverse bedrijven steeds vaker dat er serieuzer ingezet wordt op het verhogen van de kwaliteit ‘aan de voorkant’. Vanuit het testvak roepen we al lang dat juist daar grote winst te behalen valt. Recent heb ik nog een opdracht vervuld als ‘Testcoördinator Voortraject’, waarbij er expliciete aandacht (met name vanuit test) geschonken moest worden aan de specificatiefase, omdat het product daarna in een outsourcingstraject ‘verdween’ om een hele poos later weer ‘boven te komen’. Er zijn dan ook allerlei redenen te bedenken waarom juist ook daar zwaar op de kwaliteit van het ‘Voortraject’ in te zetten.

De opleiding

Maar goed, ik heb deze training nu dus gevolgd en het examen afgelegd. Hoe was dat nu, een driedaagse training in het opstellen van requirements volgens de IREB-standaard? Mijn belangrijkste persoonlijke doelstelling was om na de cursus beter in staat te zijn om requirements te valideren en beter inhoudelijke feedback te kunnen geven. Aan dat persoonlijke doel is zeker invulling gegeven. De cursus is gebaseerd

op de IREB-syllabus. Die geeft aan wat er binnen de context van de cursus besproken moet worden en wat er als bekend verondersteld wordt, voorbereidend op het examen en dus het kunnen behalen van het IREB-certificaat. Daarnaast is de cursus flink gevuld met aanvullende informatie, welke niet (altijd) expliciet in de syllabus wordt vermeld, maar vooral invulling geeft aan de praktische toepasbaarheid van de theorie. Binnen de cursus wordt middels een doorlopende case het hele traject van het (in eerste instantie al brainstormend) verkrijgen van requirements tot het stapsgewijs zorgvuldiger uitwerken en uiteindelijk reviewen van de requirements doorlopen. Hierbij wordt er geoefend met o.a. het bepalen welke stakeholders een rol spelen, welke communicatietechnieken toegepast kunnen worden om de gewenste informatie helder te krijgen, wat voor type requirements er zijn en hoe die te beschrijven en ook het vastleggen van delen van de verkregen informatie middels hiervoor geschikte modelleringtechnieken. Denk hierbij aan (grotendeel op UML gebaseerde) technieken als Use Case diagrammen, State Transition-, Activity en Sequence diagrammen, Class diagrammen en Entity Relationship Models. Ook het onderwerp 'Requirement Management Tools' komt aan de orde.

Het examen

Het afsluitende examen aan het einde van de derde dag is een 75 minuten durend multiple choice examen, waarbij er veelal vragen zijn uit de categorie 'kies uit de gegeven mogelijkheden de twee juiste antwoorden' en 'gerelateerd aan de beschreven case, geef van de beschreven stellingen aan of ze juist of onjuist zijn'. In totaal bestond het examen (in mijn geval) uit 45 vragen, met daarbinnen de nodige subvragen. Het examen was door alle examenkandidaten binnen de tijd af te ronden, maar wel met het gevoel dat het flink doorwerken was.

En nu nog even afwachten of ik het certificaat behaald heb…. (ik heb er al een plekje voor gereserveerd boven mijn bed…☺)

Peter van Delft

Infotenties

SPIder groepen op LinkedIn en Plaxo

Op het internet kun je actief netwerken in de SPIder groep via zowel LinkedIn als Plaxo. De groepen zijn te vinden op de respectievelijke sites: http://www.linkedin.com/groupsDirectory en http://www.plaxo.com/groups.

NESMA

De jaarlijkse ledenvergadering van de NESMA wordt gehouden op donderdag 9 april a.s. De leden krijgen de stukken tijdig thuisgestuurd.
Het thema van de aansluitende voorjaarsconferentie is "Het meten van de effectiviteit van offshoring". Er zullen bijdragen zijn van ABNAMRO en Capgemini.
Nadere aankondigingen over het programma volgen. Aanmelding is mogelijk via www.nesma.nl

Call for Papers QA&Test Bilbao, Spanje

Op 29 t/m 31 oktober 2009 wordt alweer voor de achtste keer de internationale Quality Assurance en Test conferentie QA&Test gehouden in Bilbao, Spanje.

Het is een leuke, kleine, informele conferentie. De bezoekers (voornamelijk Spaanse bedrijven) zijn “eager to learn”. We zijn als organisatie vooral op zoek naar praktische en pragmatische verhalen (45 minuten max.) op het gebied van QA, V&V en integratie voor embedded software systemen. Ook tutorials (max. 4 uur) zijn welkom. De deadline is 6 maart 2009.

Meer info nodig? Bel of mail me of kijk op http://www.qatest.org/en/.
Wie weet, tot in Spanje!

Willem van den Biggelaar
willem@processvision.nl

SPIder Sponsor Bijeenkomst

Sponsorsessie 2008

Voor het uitvoeren van de activiteiten de Stichting SPIder worden we financieel ondersteund door een vijftal sponsoren; DNV-CIBIT, KZA, Logica, Philips en Sogeti.
Elk jaar wordt in het vierde kwartaal een zogenaamde sponsorsessie georganiseerd. Het doel van deze sessie is meerledig.

Terugblik 2008

Ten eerste een terugblik op het afgelopen jaar, 2008. Wat waren de speerpunten 2008? Welke activiteiten hebben we uitgevoerd en in hoeverre hebben we onze speerpunten bereikt? Voor 2008 hadden we de volgende speerpunten gedefinieerd:

- Publicatie werkgroep invoeringsstrategieën;
- Groei van onze conferentie;
- Nieuwe website;
- Aangaan van samenwerkingsverbanden.

De conferentie 2008 was een groot succes. Een uitgebreider programma, meer deelnemers en een grotere beurs. Het belangrijkste is misschien wel de zeer goede beoordeling van de conferentie door de bezoekers. Tevens is op deze conferentie de publicatie van de werkgroep “Invoeringsstrategieën” verspreid.
Op het gebied van samenwerking hebben we in 2008 een zeer goede start gemaakt. We hebben contacten met IPMA-NL, ASL/BiSL, Agile Holland en het Agile Consortium (DSDM). De contacten met IPMA-NL en ASL/BiSL zijn het meest intensief.
Het is ons helaas niet gelukt om de nieuwe website per 1 januari 2009 te releasen. Dit zal in 2009 gerealiseerd worden.

Speerpunten 2009

Voor 2009 heeft de Stichting SPIder een aantal speerpunten gedefinieerd. De eerder genoemde website is er daar één van. Andere speerpunten is het verder versterken van de positie van SPIder en het vergroten van de ledenparticipatie. Natuurlijk

hebben we de kwaliteitslat voor onze plenaire sessies en conferentie weer een stukje hoger gelegd.

Trends

Een vast agendapunt van de sponsorsessie is het bespreken van de trends. Welke trends zien we nu in de markt en bij de individuele bedrijven? Dit blijkt altijd een bijzonder zinvol en krachtig onderdeel te zijn om topics in kaart te brengen.
De genoemde topics zijn vervolgens gerubriceerd in zeven verschillende onderwerpen

- Verkrijgen van bedrijfseconomisch inzicht in de gevolgen van kwaliteitsmaatregelen.
- Hoe vergroot je de effectiviteit en verbeter je de efficiency?
- Hoe kunnen we het samenwerken goed laten functioneren?
- De rol van productkwaliteit.
- Verbeteren volgens de holistische benadering.
- Wat zijn de gevolgen van de crisis voor het vakgebied kwaliteit en process improvement?
- De rol van metrieken bij dit alles.

Hoe verder?

Deze maand zijn we onze speerpunten t.o.v. de genoemde trends aan het herijken. Wat betekenen de trends nu voor ons speerpunten, voor onze plenaire sessie en voor onze conferentie? Welke trends nemen pakken we in 2009 op en waarom?
Hoe het ook zij, het belooft een bijzonder interessant jaar te gaan worden met goede plenaire sessies en een zeer aantrekkelijke conferentie!

Tenslotte

Tenslotte wil ik onze sponsoren bedanken en kijk uit naar de samenwerking in 2009.

Jeroen Macke
Bestuur stichting SPIder

Kalender

De evenementenkalender bevat een overzicht van internationale conferenties op het gebied van SPI, metrieken en softwareproductkwaliteit. Daarnaast zijn de activiteiten van SPIder opgenomen.

Ook nationale evenementen op het gebied van softwareproduct- en procesverbetering kunnen in deze evenementenkalender worden opgenomen. Via de SPIder Koerier kan een organisator van SPI gerelateerde evenementen een selecte groep van geïnteresseerden bereiken. Voor commerciële evenementen zoals conferenties, workshops, lezingen en andersoortige bijeenkomsten vraagt de redactie een kleine bijdrage in de kosten.

2009	
4 maart	SPIder plenaire sessie Ⓢ
24-27 maart	13th European Conference on Software Maintenance and Reengineering, Kaiserslautern Germany, www.csmr.eu
5 apr.	SPA 2009: Software Practice Advancement, Bedfordshire, England www.spaconference.org/spa2009/index.php
9 apr.	NESMA Voorjaarscongres 2009
15 mei	Deadline artikelen voor de SPIder Koerier Ⓢ
27 juni	Nieuwe maakbaarheid - Over maken van organisaties, veranderingen en maatschappelijke vernieuwingen Verandermanagement congres georganiseerd door Sioo e.a. voor M&O (Management & Organisatie) Kosten: 495 euro (395 euro voor leden M&O)
14-17 sep.	Joined WICSA en ECSA (software architecture) http://www.wicsa.net/
6 okt.	SPIder Conferentie 2009, Ede Ⓢ
12 nov.	NESMA Najaarscongres 2009

Ⓢ = SPIder event
✓ = korting voor SPIder donateurs

SPIder Werkgroepen

Op dit moment zijn er binnen SPIder de volgende werkgroepen actief:

- SPI Invoeringsstrategieën, voorzitter André Heijstek
- Roadmaps, voorzitter Jan Jaap Cannegieter
- Requirements, voorzitter Arno van Herk

De SPIder stichting faciliteert haar leden om in werkgroepen activiteiten op te pakken. Onderwerpen voor werkgroepen zijn welkom en kunnen worden ingebracht bij het bestuur.

SPIder Werkgroep: Requirements

De leden van de werkgroep Requirements gaan onverdroten verder met verdieping van diverse onderwerpen op dit gebied. Op dit moment bestaat de groep uit 35 actieve deelnemers. Na de plenaire sessie eind mei 2008, zijn nieuwe onderwerpen geïdentificeerd waarover in subgroepjes met elkaar wordt gediscussieerd.
De groep “Mens” gaat zich bezig houden over de vraag hoe de weerstand die op veel plaatsen toch nog leeft ten aanzien van requirements management om te buigen in verbinding.
De groep “Methoden” heeft na de presentatie op de plenaire sessie nog voldoende stof tot nadenken over methoden en processen.
De groep Traceability gaat een business case model opstellen voor het toepassen van requirements traceability, en het identificeren van relevante input parameters die het business case model beïnvloeden.

De groep Requirements Life Cycle management constateert dat lifecycle management op dit gebied nog best wel een uitdaging is. Door middel van een aantal praktijkcases wordt de problematiek geanalyseerd en getracht tot een generiek model om dit succesvol toe te gaan passen.
Requirements in Agile projecten is het onderwerp van onderzoek van de groep Requirements & Agile, waarbij vanuit het Agile Consortium Benelux het verzoek is gekomen voor enige samenwerking op dit gebied!
Over use cases en requirements wordt veel gediscussieerd, de groep Use Cases gaat zich hier in verdiepen.
Een nieuwe term die het laatste jaar in de wetenschappelijke wereld op is gekomen, is “goal-driven requirements engineering”. De gelijknamige groep verdiept zich hierin om een beeld te vormen van wat dit inhoudt en welke nieuwe ideeën hieruit voortvloeien.

Duidelijk is dat requirements een enorm breed gebied bestrijken waar nog het nodige in te ontdekken valt. De omvang van de werkgroep blijft onveranderd groot, wat een indicatie is voor het feit dat het onderwerp zeer actueel is en veel mensen bezighoudt. De individuele groepjes gaan de komende maanden met elkaar zich verder verdiepen in hun onderwerp. Op 9 juni staat weer een ‘grote’ bijeenkomst van de werkgroep op de rol, waar we dan met elkaar de ervaringen en bevindingen delen.

De SPIder Organisatie

SPIder is de Nederlandse netwerkorganisatie voor SPI. SPIder organiseert jaarlijks een conferentie, minstens drie plenaire sessies met sprekers met variërende thema’s. In aparte werkgroepen worden thema’s bediscussieerd en verder uitgewerkt. De SPIder Koerier is het medium dat minstens vier maal per jaar verschijnt. Hierin kunnen lezers hun mening uiteenzetten en interessante ervaringen delen met SPIder leden. SPIder is een stichting, non-profit organisatie, welke wordt bestuurd door vrijwilligers.

Lidmaatschap van SPIder is gratis en alle activiteiten, met uitzondering van de conferentie, zijn gratis toegankelijk. Dit wordt mede mogelijk gemaakt door onze reguliere sponsors. Dus dank voor de bijdrage van Philips, Sogeti, KZA, Logica en DNV-CIBIT!

Ook kent SPIder donateurs, zowel bedrijfsmatig als individueel. Donateurs hebben bij ons een streepje voor, en krijgen extra voordeel op de activiteiten van SPIder en van onze zustervereniging. Wil je SPIder ook steunen, meld je dan aan bij het SPIder secretariaat.

Het SPIder bestuur bestaat uit de volgende personen:

- Jeroen Macke, voorzitter
- Niek Pluijmert, penningmeester
- Martin Muller, strategie, donateurs, Q Society
- Kasia Wiacek, plenaire sessies
- Wil Leeuwis, website, SPIder conferentie
- René Krikhaar, SPIder conferentie, SPIder Koerier, nieuwsbrieven

Informatie over SPIder is te vinden op de website: www.st-SPIder.nl

Voor reacties en bijdragen op de ***SPIder website*** kunt u zich richten tot:
Redactie SPIder web, Wil Leeuwis
E-mail: w.leeuwis@gmail.com

Colofon

De inhoud van de SPIder Koerier wordt verzorgd door het SPIder netwerk. Dat betekent dat de redactie met plezier artikelen ontvangt voor publicatie. Goede ideeën om op andere wijze zinvolle inhoud te geven aan de Koerier zijn welkom: een speciale reeks van artikelen rondom een bepaald thema, een discussie via de Koerier in de vorm van meningen en opinies of een column. Je ideeën zijn welkom, we kunnen ze bespreken tijdens de conferentie, per e-mail of telefoon.

De SPIder Koerier wordt gemaakt onder auspiciën van SPIder door René Krikhaar en Cantrijn Secretariaten. Voor artikelen, mededelingen, reacties en vragen m.b.t. de ***SPIder Koerier*** kunt u zich wenden tot:

SPIder Koerier
E-mail: koerier@st-SPIder.nl

Volgende deadline van de SPIder Koerier is 15 mei 2009.